# 5.0 Intelligent Software Agents

## Introduction and Definition

The notion of software agents has been with us for a long time. Certainly since the formative years of artificial intelligence, designers have flirted with the notion computers might one day have a degree of apparent independence that would warrant their being called an agent of the user. But imbuing a machine with the attributes of an agent has been difficult to achieve. Not until processor power began to exceed appreciably that needed by the application at hand, have resources been available locally to give the user this type of support. Now that has changed. With evermore affordable processing power available at the user's first point of interaction to this new distributed computing world, there is the luxury of giving the machine attributes that, to the user, seem autonomous and intelligent. Whether they are either, of course, is in the eye of the beholder. Semantically, an agent is simply one authorized to work on behalf of or as a representative of another. In computer science research, one expects an "intelligent" software agent to use reasoning and persistence in performing its assigned task. Other, more human-like attributes such as trust are more controversial.

The lack of a precise definition of a software agent, unfortunately, gives the developer and marketer wide latitude in just how much or how little functionality is present. At their best, software agents are capable of representing a user or owner in the accomplishment of specified tasks without his or her having to prescribe or even be aware of how it is to be done. That degree of detachment has some influence on whether the user considers the agent intelligent or not. The more computer-aware the user, the more reasoning power and autonomy the agent must have to be termed intelligent.

In the development of agents there may be a propensity to ascribe human attributes to a program as its functionality increases. Because computers are the first machines that can take on tasking through abstract, human-compatible language, there is some reason for this tendency. But care must be given in not misleading users as to the reasoning power and adaptability software programs like agents actually have. It is doubtful, however, enough precision will emerge in describing such capabilities, so imputing functionality not actually present will be an ongoing problem. Because of this difficulty, the term agent has mixed acceptance by many experts. That reticence will not likely prevent it from being more popularized.

## 5.1 Principal Motivations

Why are software agents important in the evolution of computing? Listed in Table III are some of the reasons why agents will be of increasing importance in dealing with a complex world of distributed information and resources the normal user will find confusing and threatening.

*Table III Motivations for the Use of Software Agents*

- The quantity of information will be too vast and its quality too uneven for most humans to suffer through.
- The locations of desired information are too broad and nonintuitive.
- An increased "flatness" (lack of hierarchy) in the world of sources and sinks for information makes dealing with it less understandable. This gets emphasized in future peer-to-peer systems.
- Data and database heterogeneity demand a variety of translators.
- Stored media will have increasing dimensionality in various formats.
- The notion of delegation, broadly defined, will become more available.
- A broadly accepted commercial infrastructure for on-line information will provide a more consistent interface.
- The need to hide complexity.
- The need for new programming models for a distributed computing environment.
- To need for improved human-computer interaction.

## 5.2 Important Counter-Pressures

Even though the above motivations seem to make agents inevitable, there are near term obstacles. Some are due to the difficulty of the technology itself while some are due to the difficulty in vendors or users being able or wanting to organize themselves.

*Table IV Obstacles in the Development and Use of Software Agents*

- A perception that agents pose a risk as they masquerade an owner's malintent.
- The lack of a formal framework for a trusted agent. Will it perform as specified?
- "Training" agents is difficult. Can they be trained to do only prescribed tasks in a remote setting the user may not know or understand?
- The lack of a common language - Too many conventions, not enough translators (scripting and interpreter languages are the most favored at present).
- The absence of commonly accepted models of reasoning or negotiating.

## 5.3 Functionalities, Types, and Models

Agents are clearly enjoying wide discussion in the *user community*. Given that and the above background, it is perhaps best to begin a discussion of agents using some functional examples. If the popularity of the concept continues, there will be countless versions of them, but here only four will be considered:

- Advisory agents - Those agents able to monitor a situation and give feedback with or without recommendations. Generally, application-specific. Monitoring, at some level of sophistication or abstraction, should be an attribute of all agents.

- Personal assistants - Most likely to appear as adjuncts to human-computer interaction (HCI). Will offer assistance in specifiable tasks.
- Traveling (Internet) agents - Roving, mission-specific, with broad awareness and interface potential. [Other than some disk access, most all web retrievals now run entirely on the user machine. Where information gathering processes run is ultimately a matter of money and risk.]
- Multiple Collaborating Agents - Multiple agents with some common goals, of varying sophistication; may be physically or logically separated.

Another, and perhaps more general way to describe intelligent agents is by their degree of intelligence. Discussing the amount of intelligence in anything carries both difficulty and controversy. If agents are an inevitable direction in software development, we must have a way to discuss them. Here are three types of intelligent agents:

- Directed-Action Agents - Has fixed goals but can react to the data and the environment it encounters as long as they have been explicitly anticipated. Little reasoning ability except for self recovery. Might be termed a “do-this” agent.
- Reasoned-Action Agents - Has fixed goals and is able to monitor other objects (data and processes). It can reason about what it encounters and take alternative action. Being able to reason implies that it has both a knowledge base (data and rules) and some processes to use. Might be termed an “achieve-this” agent.
- Learned-Action Agents - Has the above capabilities plus it can accept more general goals and is capable of altering or adding to them under guidelines. Has broad awareness of its environment, the data it encounters, and, importantly, itself. Might be termed an “accept-this” agent.

Whether or not software agents will catch on in the *development community* is in great measure determined by the underlying models on which they are constructed. Attributing intelligence to agents means they will likely be built on the foundations of AI. Agent models are often based on concepts such as perception/action, belief/desire/intent, expert systems, game theoretic, and others. AI programs are widely embedded in existing software and AI will continue to provide the foundations for new capabilities. The technology of distributed computing will also be a necessary component. Both are needed over the long term. (See Artificial Intelligence, Chapter 7.)

Lastly, agents will become so much a part of the HCI that it will be difficult to separate the two fields. Agents will almost certainly play a role in the next step in HCI as applications programs give way to a more integrated, task-oriented type of computer-based work. Another obvious need for agents, however, will be in powerful but ultra-portable computers (e.g., PDAs) where the more traditional input/output modalities such as keyboards are not available.

## 5.4 Realizations in Software Agents Within 10 Years

Over the next decade all of the above types of agents will be introduced and in common use at some level of sophistication. The rapid growth of the Internet may draw the most attention and thus the *traveling agent* may get broadly developed first. Agents that do appreciable

remote execution will be the most difficult to “host” or “serve”. The term itself suggests a warning to some but the acceptance of someone else’s code in your machine is already widely practiced under not-so-pointed terminology. WWW home pages, for example, are executable codes that the user invites in now. This is clearly an issue of security and therefore the initiative towards encryption and other solutions now underway on the Internet will continue with emphasis if business is to thrive there.

User or system assessment and characterization by a local *advisory agent* in a carefully circumscribed domain will be available. A good example of an advisory agent is its use in an instructional setting. Here an agent will start with a basic set of instructional goals, add an ability to monitor where the student is in relation to those goals, and then set the course of instruction issuing feedback, including recommendations, along the way. A natural evolution in the sophistication of situation assessment and how it is presented will make the agent appear more intelligent and interactive.

The *personal assistant agent* will hit the marketplace in 1996 with Apple’s new operating system. These agents will assist the user in specifiable tasks such as electronic mail, calendars, conferencing set-up, object search, and so on. These agents, unlike the simple macro-recording agents of HP’s New Wave, are reputed to have some ability to reason about what they are “seeing” and to act according to some prescribed user guidelines.

An important question concerning multiple *collaborating agents* is the level of complexity of a given agent. It is doubtful that small elemental agents can aggregate into something powerful or “intelligent.” So, the question is what should the atomic level of an agent be such that its contribution can aggregate toward coherent, integrated behavior? That is undoubtedly task and situation dependent, but to date there has been no unexpected or superior behavior from such aggregations. One level of agent granularity is shown in SRI’s Open Agent Architecture depicted in Figure 5.

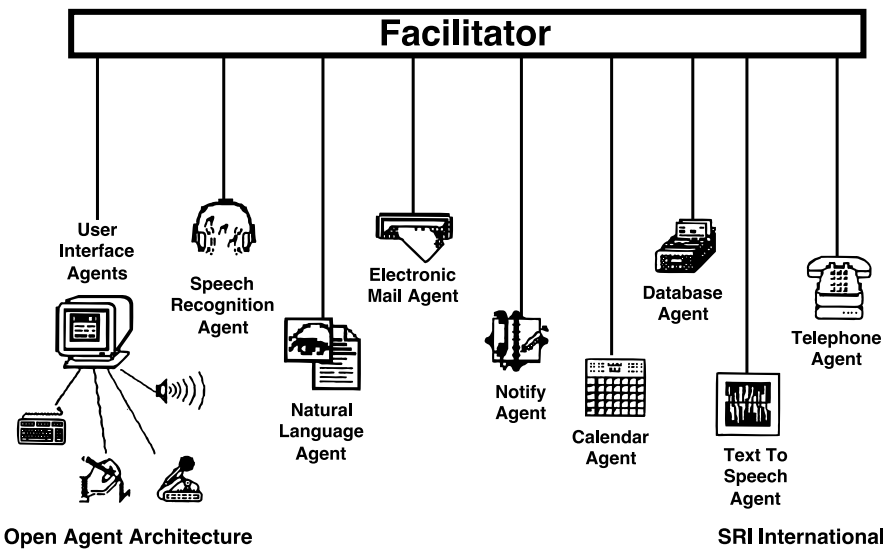


*Figure 5 SRI’s Open Agent Architecture*

One area of development that could accelerate the use of agents in a wide range of settings is agent language; that is, a common agent language or environment containing user-agent, agent-agent, and agent-host (server) interaction. Several procedural, interpretive, and declarative (e.g., ACL/KIF/KQML) languages are already in use in new companies or in universities. Reducing agent functionality to a simple common form should, as in the reasons related to object-oriented software, simplify language and interactions and at the same time make trusted interaction more likely. Network-based operating systems that provide a homogeneous docking interface may also act to protect both host and agent. Common, public rules for agents that promote desirable inter-agent and host-agent behavior will be forthcoming. Scripting languages having just this purpose are now in use.

Regarding the underpinnings or catalysts to the growth of agents, several areas are worth noting. These will be expressed in human-like attributes even though the capabilities will be far from human. This illustrates the strong need for commonly accepted terms to describe agent attributes. Some agent properties:

- Veracity - Some availability of trust through tightly scripted interface language and cryptographic authentication.
- Competence - Some capacity for accurate observation and interpretation, limited by cost, to construct, maintain, and operate agents.
- Persistence - Recovery using some form of reasoning.
- Security and Safety - Some assurance through host constraints and agent veracity measures above. But two strategies will have to be avoided: 1) broad or intricate activities carried on by an agent in a host machine and 2) any true universal solution not supported by cryptography. [A worthy attribute: An agent's design is made such that its behavior from a host's perspective gets rewarded when it conforms exactly to its advertised purpose and punished, perhaps annihilated, when it does not.]
- Autonomy - In traveling agents, some "en route" decisions using reasoning. Autonomy is largely the province of the reasoning and learning agents; that is, being able to assess the situation and take alternative action and, in the latter case, remembering to avoid it next time.

Some applications of software agents are under development or test and illustrate the directions future design will take:

- Open Agent Architecture - Implements multimodal, distributed HCI (SRI)
- OASIS - An air traffic control system under test in Australia (AAII)
- FLiPSiDE - A "blackboard system" for agent interaction (Stanford)
- Telescript - A proprietary, commercial, general purpose scripting language (General Magic)
- SmalltalkAgents - A scripting language based on Smalltalk (Quasar)

- Tool Command Language (Tcl) - High-level, hypercard-like, machine-independent scripting language (public domain)
- Safe-Tcl - Secure version of Tcl (First Virtual).

## 5.5 Realizations in Software Agents Within 20 Years

The following projections are evolutionary, not revolutionary, and they begin with an assumption that the Internet will lead, directly or indirectly, to a global electronic information and commercial infrastructure. That infrastructure will be a consistent, universal, and pluralistic system. It will permit your personal or corporate computing environment, not necessarily local, to transparently represent you in a wide variety of transactions such as educational or learning systems, commerce (buying and selling), conferencing, scheduling, entertainment, mail, and much more. If such an infrastructure is not forthcoming, it will be for organizational and not technical reasons.

More about agent attributes:

- Veracity - Will be guaranteed through at least one of several methods: one-time authentication from a trusted third party plus checksumming; task execution or memory constraints in hosts; agent (of arbitrary complexity) surrounded by a simple shell written in script of constrained functionality; creation of very isolated environments in the host.
- Competence - Specified abilities to gain closure, accuracy.
- Autonomy - Specified circumscription but still wide latitude on agency and ability to negotiate or have volition. Agent-agent negotiations will occur under relatively simple, user-defined, and legal guidelines. Context will likely be buying and selling simple products rather than contracts in which both costs and benefits are more ambiguous.
- Delegation - The ability to receive abstract, human-language commands and carry them out in ways transparent to requester. It is not necessary to think of an agent as yours, as a single module, or as local. But roving, rogue, ownerless agents are an act of information warfare even if they have no malicious intent. Agents must always be responsible to some user!

## 5.6 Untethered Realizations in Software Agents

While twenty years seems an eternity in the computing world, there is one portrayal of the future of agents and HCI that has no time limit, only direction. This direction is from the present hands- and eyes-intensive machine toward a more amorphous system that might be called a delegatable assistant. These properties will emerge:

- Entire computers will become delegatable agents with natural language capabilities (See Human-Computer Interaction, Chapter 4.0)
- Trusted interactions will occur between users and hosts via their agents.

- Collaboration will occur among task or knowledge specific autonomous agents to achieve an integrated goal.

The notion of agents as delegatable software will occur as part of the evolutionary mainstream of computer development. Human-computer interactions will be done dominantly in human language terms much in the manner of requests or delegations. Keyboards will survive for text-intensive input.